22 | RASP：写规则写得烦了？尝试一下更底层的IDS

2020-02-05 何为舟

讲述：何为舟

时长 13:00 大小 10.43M

你好，我是何为舟。

在前面的课程中，我们已经介绍了防火墙、WAF 和入侵检测。这些产品都有一个共同的特性，就是基于网络请求或者系统行为对攻击的特征进行检测，然后再采取相应的防控手段。这些安全产品基本都和应用本身解耦。也就是说，基本上我们不需要对应用做任何开发和改动，就能够部署这些安全产品。

尽管解耦在部署上能够节省很大的成本，但解耦同样意味着，安全产品和应用本身是通过端口、请求等形式来进行数据交换的。换一句话说，安全产品只能够看到应用输入和输出的数据，并不知道数据在应用内的流动情况。因此，这种工作模式不可避免会产生一定的误判和漏报。

我们来看一个关于 WAF 检测 SQL 注入的例子。下面是请求代码：

```
1 http://server.com/login?username=test&password=" or ""="
```

复制代码

WAF 可能会检测到 password 参数中的 SQL 注入痕迹进行拦截。如果应用采用的是安全的 PreparedStatement 方法，那这个 SQL 注入就不会生效，也就不需要拦截。但是 WAF 和应用解耦，让 WAF 不知道应用的逻辑，从而产生了误报。

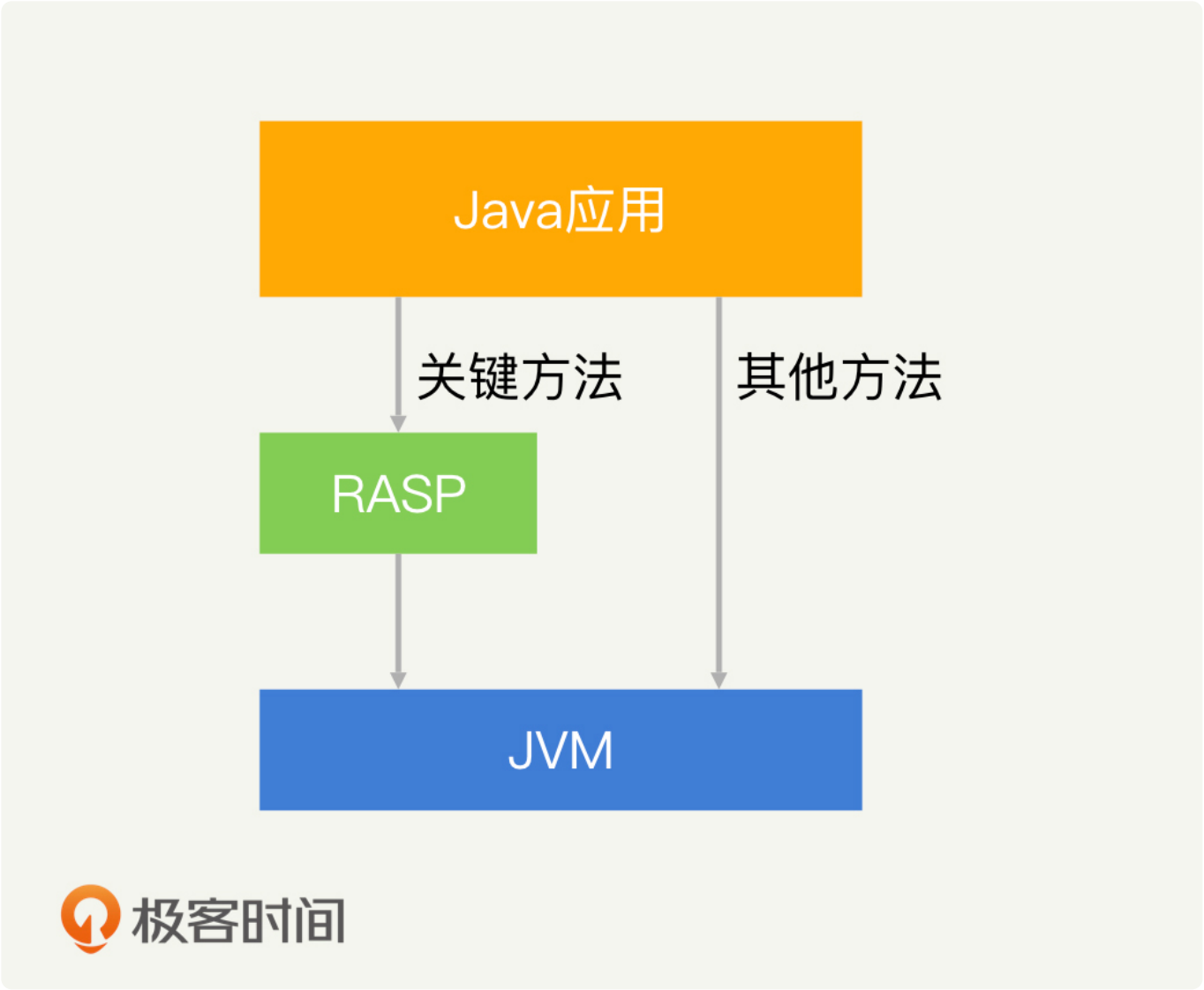
所以，对于任何安全产品来说，能获取到的数据和信息越多，检测的能力就越强，误判和漏报的概率也就越低。因此，2012 年，Gartner 提出了 RASP（Runtime Application Self Protection）的概念，就是希望将安全产品部署在应用的底层，完全站在应用的视角去发现攻击行为，从而实现更加完善的安全防护。

RASP 的原理

想要利用 RASP 实现更完善的安全防护，首先我们要知道，什么是 RASP？以及如何实现 RASP？

RASP 的设计思路是通过监控应用的底层，来从根本上发现攻击行为的产生。

以 Java 为例，Java 应用运行在 JVM 之上。因此，JVM 就是一个底层，它能够看到所有的应用信息。我们可以通过 Java Agent 的形式将 RASP 运行在 JVM 上，然后借助 Instrumentation 技术 Hook 关键的类和方法。关键类和方法具体有哪些，你可以参照 OpenRASP 的 Hook 列表。这样一来，RASP 就能关注到应用安全相关的信息和调用了。



同样的原理，在 PHP 中，我们可以通过 PHP 扩展库来实现 RASP；在.Net 中，我们可以通过 HostingStartup 来实现 RASP。

如果你想要研究 RASP 产品，那我推荐你使用百度的OpenRASP。因为 OpenRASP 在开源市场中认可度比较高，也是目前各个公司用来研究 RASP 产品的一个主要对象。

RASP 的优势和劣势

我们经常会将 RASP 和 WAF 拿来作比较，因为它们主要关注的都是应用相关的 Web 安全问题。那么对比 WAF，RASP 有哪些优势和劣势呢？

首先我们来看优势。在开头我们就提到了，RASP 对比于 WAF 最大的优势在于 RASP 运行在应用的底层，从而能够知道应用运行时的上下文（比如：用户、代码逻辑、SQL 语句等）。在 Web 安全中，我们针对 Web 安全的攻击原理进行过总结：SQL 注入、反序列化等漏洞其实都是通过输入数据，篡改应用的正常逻辑实现的攻击。

对于 WAF 来说，它只能够判断出输入的数据“可能”会篡改应用的正常逻辑，因此 WAF 的拦截决策都来源于这个可能性。而对于 RASP 来说，它知道应用的正常逻辑是什么，也知道应用接收输入后实际的逻辑是什么，如果实际逻辑和正常逻辑不一致，就必然发生了攻击。基于这种检测方式，RASP 基本不会产生误报或者漏报。

我们以 OpenRASP 防止 SQL 注入的检测逻辑为例，来看一下 RASP 是如何进行检测的。算法描述如下：

```
// SQL注入算法#1 - 匹配用户输入
// 1. 用户输入长度至少 8
// 2. 用户输入至少包含一个SQL关键词 - 即 pre_filter, [默认关闭]
// 3. 用户输入完整的出现在SQL语句中，且会导致SQL语句逻辑发生变化
sql_userinput: {
    name:        '算法1 - 用户输入匹配算法',
    action:      'block',
    min_length: 8,
    pre_filter: 'select|file|from|;',
    pre_enable: false,
    anti_detect_filter: 'add|all|alter|analyze|and|any|as|a
    anti_detect_enable: true,
    lcs_search: false,

    // 是否允许数据库管理器 - 前端直接提交SQL语句
    allow_full: true
},
```

第 1 步和第 2 步很好理解，都是 SQL 注入的基本特征。那第 3 步中的“导致 SQL 语句逻辑发生变化”，OpenRASP 要如何去识别呢？假设用户的输入是万能密码"or""="，那么，应用实际执行的 SQL 语句就是：

SELECT * FROM Users WHERE Username = “” AND Password = “ “or” ” =""

在这个 SQL 语句中，最后的几个字符都是用户的输入。为了检测语句逻辑的变化，OpenRASP 将这个 SQL 语句进行了 Token 化。所谓“Token 化”，就是对 SQL 语句中

的关键词进行拆解，拆解后分别是：SELECT、 *、FROM、Users、WHER、Username、=、""、AND、Password、=、""、or、""、=、""。我们能够清楚的看到，用户的输入“"“or “” =”"” 占据了 5 个 Token。而正常情况下，用户的输入应该只占据 1 个 Token。因此，只要用户的输入大于 1 个 Token，就说明 SQL 语句逻辑存在变化，我们就可以断定存在 SQL 注入了。

其次，RASP 能够防范未知的攻击。对于 SQL 注入来说，它的注入点可以是某个 GET 参数、某个 POST 的 Body、某个 Header 字段等，具体的攻击方式也多种多样：盲注、基于 Insert 的注入等。

WAF 的检测规则是一个一个去覆盖这些攻击点和攻击方式。如果黑客发现了某个新的攻击点或者使用了新的攻击方式，WAF 根本无法检测出来。

对于 RASP 来说，它实际上不关注具体的攻击点和攻击方式是什么，因为 SQL 注入攻击，最终都会使 SQL 语句 Token 化后的长度发生改变。因此，RASP 只需要判断执行的 SQL 语句 Token 化后的长度即可。所以我才说，RASP 能够有效地防御未知的攻击。

最后，我认为 RASP 还有一个比较特别的好处，就是基本不用维护规则。我前面讲过的各类安全产品：防火墙、WAF 和入侵检测，它们的本质都是检测攻击行为。而这些安全产品的检测方式不论是签名匹配、正则匹配还是行为分析，都包含了大量的规则和算法。这些规则和算法带来的最大问题，就是针对每一家公司我们都需要制定一套单独的规则和算法。因为每家公司的应用和网络环境都不一样，面临的攻击也不一样。

随着公司的发展以及黑客的对抗升级，我们还需要不停地更新和维护这套规则和算法，这就带来了极大的运营成本。而对于 RASP 来说，它当然也需要规则和算法来支持，但是它的规则和算法相对统一。

比如，在 Java 中，不论你是使用的哪种开发框架，最终执行 SQL 语句的都是底层的 JDBC 插件，在这个层次上，攻击的特征都是一致的。因此，RASP 基本只需要维护一套统一的规则和策略，就能够适用于所有的公司和应用了。所以，百度在 OpenRASP 覆盖面说明中敢说“若发现不能拦截的攻击或者误报的情况，请联系我们”，就是这个原因。开源的 WAF 只会提供一个维护规则的入口，而不会帮助你进行维护。

尽管 RASP 存在许多明显的优势，但是目前来看，国内对于 RASP 的实际落地，普遍还在试验阶段，我很少见到 RASP 在公司范围内大规模推广落地案例。这是为什么呢？要搞清楚这个问题，就不得不提到 RASP 的劣势了。RASP 的劣势主要有三方面，下面，我们一一来看。

我认为最大的劣势在于推广难度上。尽管我们一直在提安全，但是事实上，大部分的开发人员并不认可安全，他们也不接受任何可能对应用产生影响的安全产品。这是因为，这些安全产品增加了检测的逻辑，就必然会影响应用的正常运行。而且，WAF 等拦截性安全产品产生的误报，会让正常的业务请求受到影响。

但是，防火墙、WAF 和入侵检测，实际上都已经在各个公司很好地落地了。我认为这都归功于安全人员或者运维人员的“强推”。

在部署一款 WAF 的过程中，实际上是不需要开发人员参与的，运维人员在网关上直接部署就可以了。而 RASP 不一样，RASP 和应用强耦合，它需要由开发人员去部署。比如，Java 中需要通过命令 java -javaagent:rasp.jar -jar app.jar 来启动 RASP，其中的参数 javaagent 只能由开发人员进行配置。因此，RASP 的推广实际上是安全意识的推广，所以难度也比较高。

其次，RASP 的解决方案并不通用。从语言支持上来看，目前 RASP 只在 Java、PHP 和.Net 语言中具备成熟的产品。其他高级语言，如 Python 等，可能是因为没有很好的 Hook 方案，所以目前仍然局限于研究阶段。这也是 RASP 强耦合所带来的弊端：每一种开发语言，甚至是语言下不同的开发框架，都可能会需要一套独立的 RASP 产品。而 WAF 等安全产品，因为网络和系统都比较统一，则不受此限制。

最后，RASP 在性能问题上也备受争议。尽管目前成熟的 RASP 产品宣称它的性能影响已经低于 5%，甚至更低了，但在实际落地的过程中，确实会出现因为系统和应用的差异，而导致性能恶化比较严重的情况。这也是 RASP 在兼容性不足上所表现出来的缺陷。

RASP 的其他功能

除了针对应用的攻击行为进行检测和拦截，和 WAF 一样，RASP 也能够提供很多额外的安全防护功能。

毫不夸张地说，WAF 能实现的功能，RASP 都能够实现。因此，WAF 中的数据保护、虚拟补丁等功能，RASP 也都能够提供，原理也是一致的：都是通过拦截并修改 HTTP 请求和响应，在 HTTP 内容中加入额外的安全特性，比如 Cookie 加密。

除此之外，因为 RASP 部署于应用的底层，知道应用的全部信息，所以它本身可以对应用的安全性进行评估。最简单的评估问题如下：

是否使用 ROOT 权限运行了应用；

在连接数据库的时候，是否使用了弱密码；

使用了哪些插件，插件是否包含漏洞。

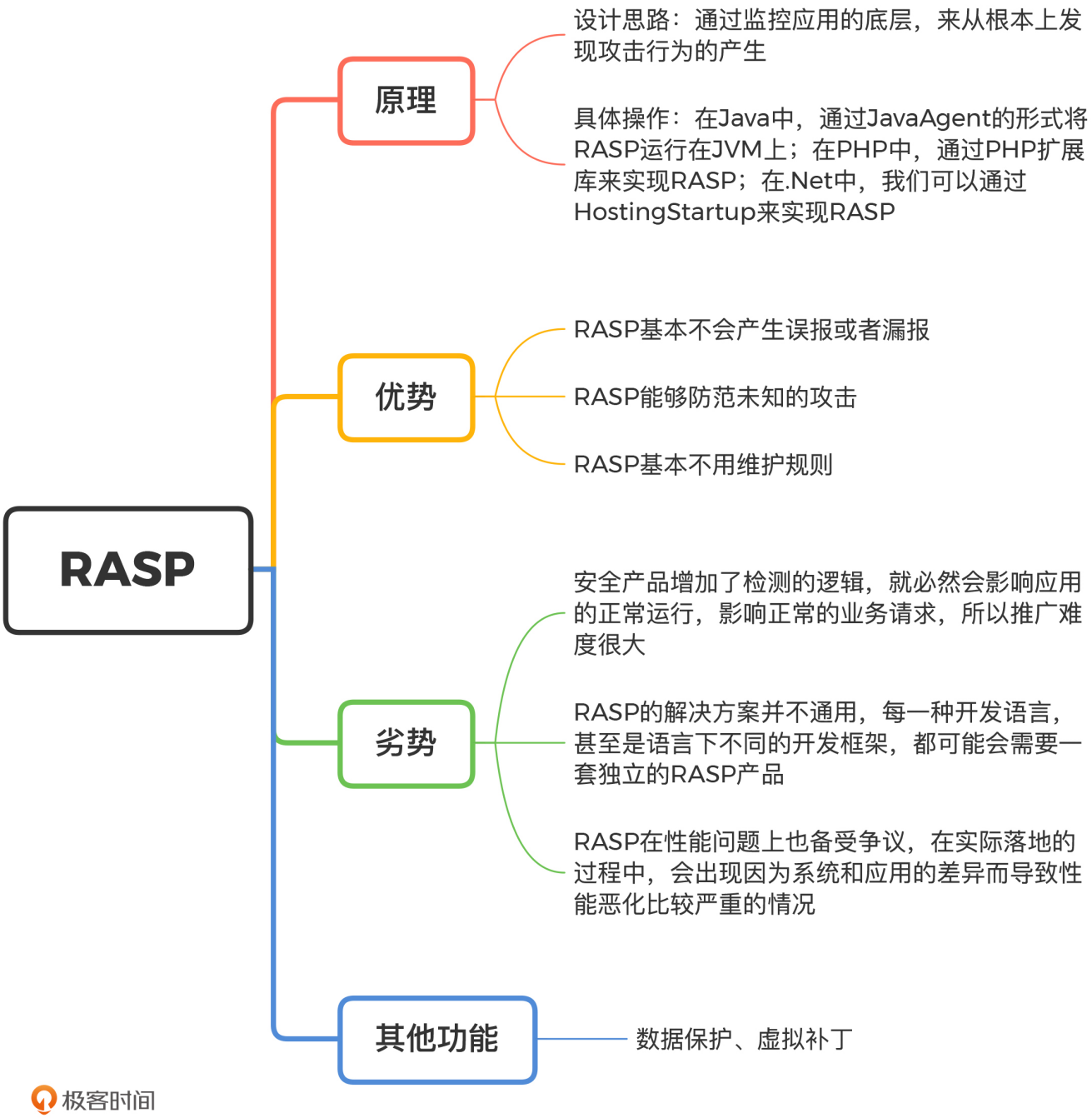
RASP 可以在应用运行的过程中对这些高危的进程操作进行检测，从而提醒你采取更安全的实现方式。

总结

好了，今天的内容差不多了，我们一起来总结一下，你需要掌握的重点内容。

RASP 的功能确实能给应用的安全性带来一个质的提升。对比低耦合的 WAF 等安全产品，RASP 的准确率、覆盖度都有较大优势。但是，正因为耦合度过高，RASP 部署起来比较麻烦，实际推广落地的时候，经常出现开发人员不配合的情况。总的来说，推广的难度属于管理上的问题，需要你想办法说服开发人员。单纯从安全角度上来说，我认为 RASP 是一款提升应用安全性的最佳安全产品。

另外，我认为在实际工作中，我们也可以将 RASP 和其他安全产品进行适当 组合，以达到取长补短的目的。比如说，RASP 推广比较难，我们可以只作局部的部署。这些局部的部署可以当作 WAF 的样本数据来源，只要 RASP 在底层发现了攻击行为，就将相应的表层特征输出到 WAF。这样一来，我们就可以将 RASP 在局部上的防御能力拓展到整体的 WAF 上，从而实现全面的安全防护提升。



思考题

最后，给你留一道思考题。

在这一讲中，我们只是以 SQL 注入为例，讲述了 RASP 是如何进行攻击检测的。你可以思考一下，对于其他的攻击方式，如反序列化漏洞、命令执行和 SSRF 等，RASP 又该如何防护呢？你可以先学习一下 OpenRASP 的说明文档，之后再来思考这个问题。

欢迎留言和我分享你的思考和疑惑，也欢迎你把文章分享给你的朋友。我们下一讲再见！

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　21 | IDS：当黑客绕过了防火墙，你该如何发现?

下一篇　23 | SIEM：一个人管理好几个安全工具，如何高效运营?

精选留言 (1)

写留言

leslie
2020-02-07

提一下课中的内容：以及对于课程对于RASP的理解；
1.目前其实是暂时处于初期阶段，即我们所说的软件1.0，其后续的许多支持暂时并未出现
2.RASP目前最大的问题在于对于编程语言的支持力度不够，尤其分布式中用的最多的PYTHON和GO语言尚未支持
3.它的部署不是有Ops去做：开发团队本身就能力高低差异巨大，开发又恰恰是最不懂这...
展开